Package 'prettyunits'

January 24, 2020

Title Pretty, Human Readable Formatting of Quantities

Version 1.1.1

Author Gabor Csardi

Maintainer Gabor Csardi `<csardi.gabor@gmail.com>`

Description Pretty, human readable formatting of quantities.
Time intervals: '1337000' -> '15d 11h 23m 20s'.
Vague time intervals: '2674000' -> 'about a month ago'.
Bytes: '1337' -> '1.34 kB'.

License MIT + file LICENSE

LazyData true

URL `https://github.com/gaborcsardi/prettyunits`

BugReports `https://github.com/gaborcsardi/prettyunits/issues`

Suggests codetools, covr, testthat

RoxygenNote 7.0.2

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2020-01-24 06:50:07 UTC

R topics documented:

prettyunits *Prettier formatting of quantities*

Description

Prettier formatting of quantities

pretty_bytes *Bytes in a human readable string*

Description

Use `pretty_bytes()` to format bytes. `compute_bytes()` is the underlying engine that may be useful for custom formatting.

Usage

```
pretty_bytes(bytes, style = c("default", "nopad", "6"))

compute_bytes(bytes, smallest_unit = "B")
```

Arguments

`bytes`	Numeric vector, number of bytes.
`style`	Formatting style: • `"default"` is the original `pretty_bytes` formatting, and it always pads the output, so that all vector elements are of the same width, • `"nopad"` is similar, but does not pad the output, • `"6"` always uses 6 characters, The `"6"` style is useful if it is important that the output always has the same width (number of characters), e.g. in progress bars. See some examples below.
`smallest_unit`	A character scalar, the smallest unit to use.

Value

Character vector, the formatted sizes. For `compute_bytes`, a data frame with columns `amount`, `unit`, `negative`.

Examples

```
bytes <- c(1337, 133337, 13333337, 1333333337, 133333333337)
pretty_bytes(bytes)
pretty_bytes(bytes, style = "nopad")
pretty_bytes(bytes, style = "6")
```

`pretty_dt` *Pretty formatting of time intervals (difftime objects)*

Description

Pretty formatting of time intervals (difftime objects)

Usage

```
pretty_dt(dt, compact = FALSE)
```

Arguments

`dt`	A `difftime` object, a vector of time differences.
`compact`	If true, then only the first non-zero unit is used. See examples below.

Value

Character vector of formatted time intervals.

See Also

Other time: `pretty_ms()`, `pretty_sec()`

Examples

```
pretty_dt(as.difftime(1000, units = "secs"))
pretty_dt(as.difftime(0, units = "secs"))
```

`pretty_ms` *Pretty formatting of milliseconds*

Description

Pretty formatting of milliseconds

Usage

```
pretty_ms(ms, compact = FALSE)
```

Arguments

`ms`	Numeric vector of milliseconds
`compact`	If true, then only the first non-zero unit is used. See examples below.

Value

Character vector of formatted time intervals.

See Also

Other time: `pretty_dt()`, `pretty_sec()`

Examples

```
pretty_ms(c(1337, 13370, 133700, 1337000, 1337000000))

pretty_ms(c(1337, 13370, 133700, 1337000, 1337000000),
          compact = TRUE)
```

`pretty_sec` *Pretty formatting of seconds*

Description

Pretty formatting of seconds

Usage

```
pretty_sec(sec, compact = FALSE)
```

Arguments

`sec`	Numeric vector of seconds.
`compact`	If true, then only the first non-zero unit is used. See examples below.

Value

Character vector of formatted time intervals.

See Also

Other time: `pretty_dt()`, `pretty_ms()`

Examples

```
pretty_sec(c(1337, 13370, 133700, 1337000, 13370000))

pretty_sec(c(1337, 13370, 133700, 1337000, 13370000),
           compact = TRUE)
```

`time_ago` *Human readable format of the time interval since a time point*

Description

It calls `vague_dt` to do the actual formatting.

Usage

```
time_ago(date, format = c("default", "short", "terse"))
```

Arguments

`date`	Date(s), `as.POSIXct` will be called on them.
`format`	Format, currently available formats are: ‘default’, ‘short’, ‘terse’. See examples below.

Value

Character vector of the formatted time intervals.

Examples

```
now <- Sys.time()

time_ago(now)
time_ago(now - as.difftime(30, units = "secs"))
time_ago(now - as.difftime(14, units = "mins"))
time_ago(now - as.difftime(5, units = "hours"))
time_ago(now - as.difftime(25, units = "hours"))
time_ago(now - as.difftime(5, units = "days"))
time_ago(now - as.difftime(30, units = "days"))
time_ago(now - as.difftime(365, units = "days"))
time_ago(now - as.difftime(365 * 10, units = "days"))

## Short format
time_ago(format = "short", now)
time_ago(format = "short", now - as.difftime(30, units = "secs"))
time_ago(format = "short", now - as.difftime(14, units = "mins"))
time_ago(format = "short", now - as.difftime(5, units = "hours"))
time_ago(format = "short", now - as.difftime(25, units = "hours"))
time_ago(format = "short", now - as.difftime(5, units = "days"))
time_ago(format = "short", now - as.difftime(30, units = "days"))
time_ago(format = "short", now - as.difftime(365, units = "days"))
time_ago(format = "short", now - as.difftime(365 * 10, units = "days"))

## Even shorter, terse format, (almost always) exactly 3 characters wide
time_ago(format = "terse", now)
time_ago(format = "terse", now - as.difftime(30, units = "secs"))
time_ago(format = "terse", now - as.difftime(14, units = "mins"))
time_ago(format = "terse", now - as.difftime(5, units = "hours"))
time_ago(format = "terse", now - as.difftime(25, units = "hours"))
time_ago(format = "terse", now - as.difftime(5, units = "days"))
time_ago(format = "terse", now - as.difftime(30, units = "days"))
time_ago(format = "terse", now - as.difftime(365, units = "days"))
time_ago(format = "terse", now - as.difftime(365 * 10, units = "days"))
```

`vague_dt` *Human readable format of a time interval*

Description

Human readable format of a time interval

Usage

```
vague_dt(dt, format = c("default", "short", "terse"))
```

Arguments

`dt`	A `difftime` object, the time interval(s).
`format`	Format, currently available formats are: ‘default’, ‘short’, ‘terse’. See examples below.

Value

Character vector of the formatted time intervals.

Examples

```
vague_dt(as.difftime(30, units = "secs"))
vague_dt(as.difftime(14, units = "mins"))
vague_dt(as.difftime(5, units = "hours"))
vague_dt(as.difftime(25, units = "hours"))
vague_dt(as.difftime(5, units = "days"))
vague_dt(as.difftime(30, units = "days"))
vague_dt(as.difftime(365, units = "days"))
vague_dt(as.difftime(365 * 10, units = "days"))

## Short format
vague_dt(format = "short", as.difftime(30, units = "secs"))
vague_dt(format = "short", as.difftime(14, units = "mins"))
vague_dt(format = "short", as.difftime(5, units = "hours"))
vague_dt(format = "short", as.difftime(25, units = "hours"))
vague_dt(format = "short", as.difftime(5, units = "days"))
vague_dt(format = "short", as.difftime(30, units = "days"))
vague_dt(format = "short", as.difftime(365, units = "days"))
vague_dt(format = "short", as.difftime(365 * 10, units = "days"))
```

```
## Even shorter, terse format, (almost always) exactly 3 characters wide
vague_dt(format = "terse", as.difftime(30, units = "secs"))
vague_dt(format = "terse", as.difftime(14, units = "mins"))
vague_dt(format = "terse", as.difftime(5, units = "hours"))
vague_dt(format = "terse", as.difftime(25, units = "hours"))
vague_dt(format = "terse", as.difftime(5, units = "days"))
vague_dt(format = "terse", as.difftime(30, units = "days"))
vague_dt(format = "terse", as.difftime(365, units = "days"))
vague_dt(format = "terse", as.difftime(365 * 10, units = "days"))
```

Index